EXHIBIT C

EXHIBIT G

Expert Report – ’890 Invalidity Chart

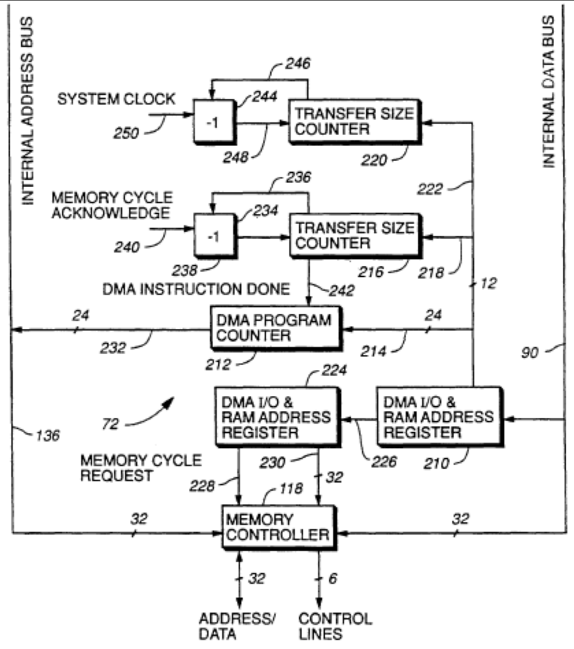
Exhibit [G]

Analysis of Transputer Architecture for U.S. Patent 5,530,890

Transputer Architecture[1] anticipates claims 11, 12, 13, 17, and 19 of the U.S. Patent No. 5,530,890 (the “’890 patent”) under at least 35 U.S.C. §§ 102(a), 102(b), and/or 102(g), and renders obvious the ’890 patent under 35 U.S.C. § 103 alone and/or in combination with other references.

Claim Language	
Claim 11	
A microprocessor, which comprises	To the extent that the preamble of Claim 11 is limiting, the Transputer Architecture discloses this limitation. For example, the Transputer Reference Manual states: The transputer is, therefore, a VLSI device with a processor, memory to store the program executed by the processor, and communication links for direct connection to other transputers. [HTCTP0111887]
a main central processing unit and	The Transputer Architecture discloses this limitation. For example, the Transputer Reference Manual states: The transputer is, therefore, a VLSI device with a processor, memory to store the program executed by the processor, and communication links for direct connection to other transputers. [HTCTP0111887] For example, as shown below, the Transputer contains a central processing unit (CPU).

[1] As disclosed by (1) The Transputer References Manual [HTCTP0111842 – HTCTP0112208]; (2) Transputer Instruction Set - A Compiler Writer's Guide [HTCTP10218078 – HTCTP10218250]; and (3) IMS T414 Transputer Product Sheet [HTCTP10218031 – HTCTP10218046].

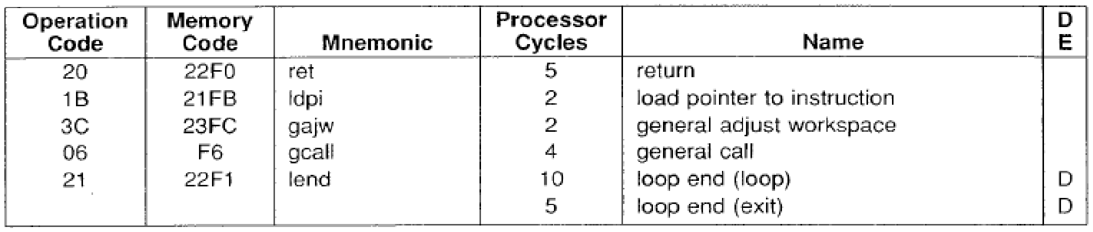
Claim Language	
	 INTERNAL ADDRESS BUS INTERNAL DATA BUS SYSTEM CLOCK TRANSFER SIZE COUNTER MEMORY CYCLE ACKNOWLEDGE TRANSFER SIZE COUNTER DMA INSTRUCTION DONE DMA PROGRAM COUNTER DMA I/O & RAM ADDRESS REGISTER DMA I/O & RAM ADDRESS REGISTER MEMORY CYCLE REQUEST MEMORY CONTROLLER ADDRESS/ DATA CONTROL LINES FIG._5 Under the Court's construction, the Transputer Architecture discloses or renders obvious this limitation. For example, the Transputer Reference Manual discloses the use of multiple Transputers in a network which communicate with each other over point to point communication links. As shown below, every Transputer has four links and each link is able to perform DMA transfers, via a controller, in both directions. Each of the Transputers has the ability to access memory and fetch and execute instructions. A transputer can be used in a single processor system or in networks to build high performance concurrent systems. A network of transputers and peripheral controllers is easily constructed using point-to-point communication.

Claim Language	
	[HTCTP0111924] Furthermore, the Transputer Reference Manual discloses a stack from which values are pushed and popped. Such a stack is also referred to as a Last-In-First-Out (LIFO) data structure. This structure provides simplicity and thus efficiency. Expressions are evaluated on the evaluation stack, and instructions refer to the stack implicitly. For example, the *add* instruction adds the top two values in the stack and places the result on the top of the stack. The use of a stack removes the need for instructions to respecify the location of their operands. [HTCTP0111890] The ’890 patent discusses this same advantage in its specification: A stack has the advantage of faster operation compared to on-chip registers by avoiding the necessity to select source and destination registers. (A math or logic operation always uses the top two stack items as source and the top of stack as destination.) [Ex. B, ’890 Patent, Col. 14, Lns. 55-57] Furthermore, to the extent this claim element requires a direct path from each register to the associated ALU input or a direct coupling between them, it would have been obvious to one skilled in the art to implement such a design. For example, Koopman (*See* HTCTP10218837 – HTCTP10219070) discloses the use of a direct path from each stack register to the associated ALU input and a direct coupling between them:

Claim Language	
	[HTCTP10218897] A person of ordinary skill in the art would have been motivated to combine the Koopman reference with the Transputer Architecture to meet this limitation if requiring a direct path or coupling between the ALU and each register because, for example, it would potentially increase performance.
an output of said arithmetic logic unit being connected to said top item register,	The Transputer Architecture discloses or renders obvious this limitation if a direct path from the ALU output to the associated register or a direct coupling between them is not required. For example, the Transputer Reference Manual discloses a system that uses a stack to store the output of the ALU. Expressions are evaluated on the evaluation stack, and instructions refer to the stack implicitly. For example, the *add* instruction adds the top two values in the stack and places the result on the top of the stack. The use of a stack removes the need for instructions to respecify the location of their operands. [HTCTP0111890] To the extent this claim element requires a direct path from the ALU output to the associated register or a direct coupling between them, it would have been obvious to one skilled in the art to implement such a design. For example, Koopman (*See* HTCTP10218837 – HTCTP10219070) discloses the use of a direct path from the ALU output to the associated register or a direct coupling between them. A person of ordinary skill in the art would have been motivated to combine the Koopman reference with the Transputer Architecture to meet this limitation because, for example, it would potentially increase performance.
said top item register also being connected to provide inputs to an internal data bus,	The Transputer Architecture discloses or renders obvious this limitation. For example, the systems disclosed by the Transputer Reference Manual all use a set of internal data buses to move data: Internally, the IMS T 414 consists of a memory, processor and

Claim Language	
counter,	Furthermore, the Transputer discloses specific instructions for loop control, which indicates the existence of a loop counter. As shown in Figure 4.15, the Transputer Reference Manual discloses a specific instruction to control looping.

Table 4.15: IMS T414 control operation codes



Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	D E
20	22F0	ret	5	return	
1B	21FB	ldpi	2	load pointer to instruction	
3C	23FC	gajw	2	general adjust workspace	
06	F6	gcall	4	general call	
21	22F1	lend	10	loop end (loop)	D
			5	loop end (exit)	D

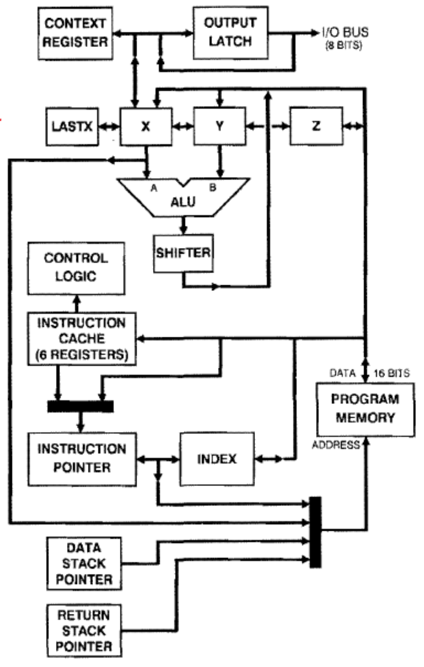
[HTCTP0111984]

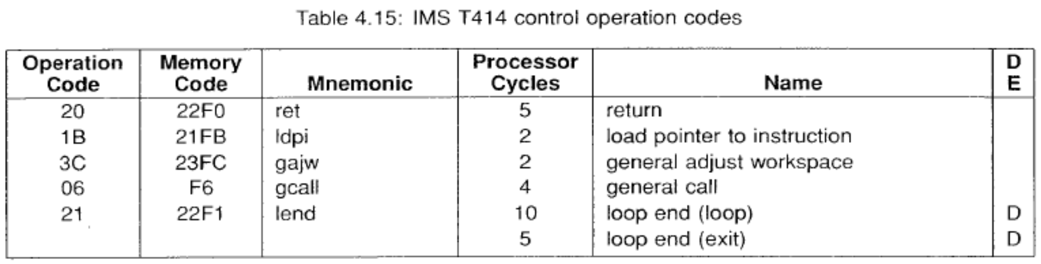
This is further explained in the Transputer Instruction Set: A Compiler Writer's Guide:

> *lend* will decrement the iteration count and, if the number of iterations remaining is greater than zero, increment the control variable and subtract **Areg** from **Iptr**. If the number of iterations left after the decrement is less than or equal to zero then execution passes to the next instruction.
>
> [HTCTP10218113]

Additionally, the A register of the Transputer can be used as a loop counter. It can be decremented by an instruction such as ADC -1 and tested using the CJ instruction. [*See, e.g.*, HTCTP10218105, HTCTP10218098].

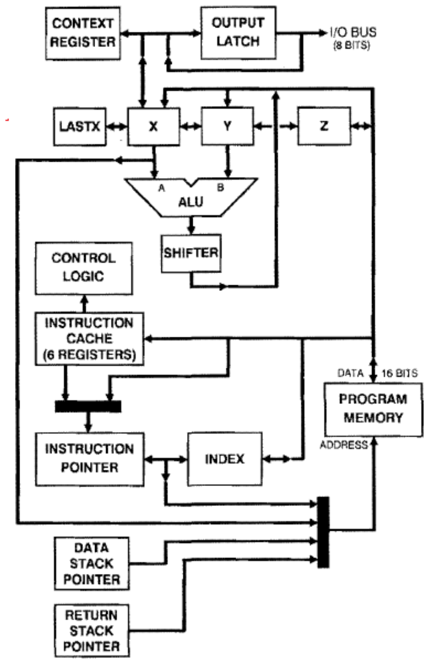
Even if this claim element requires a specific register as a loop counter, such architecture was well known by a person in the art at the time of the invention. For example, the Koopman reference discloses a specific register that may be used as a loop counter. [See, e.g., HTCTP10218898-899].

Claim Language	
	 CONTEXT REGISTER OUTPUT LATCH I/O BUS (8 BITS) LASTX X Y Z A B ALU SHIFTER CONTROL LOGIC INSTRUCTION CACHE (6 REGISTERS) DATA 16 BITS PROGRAM MEMORY ADDRESS INSTRUCTION POINTER INDEX DATA STACK POINTER RETURN STACK POINTER Fig. 4.4 — M17 block diagram. [HTCTP10218898] The top element of the Return Stack is buffered in the INDEX register. The INDEX register doubles as a count-down counter for use in program loops and the instruction repeat feature. [HTCTP10218899] A person of ordinary skill in the art would have been motivated to combine the Koopman reference with the architecture with the Transputer Architecture for this limitation because, for example, it would potentially increase the efficiency of loop execution.

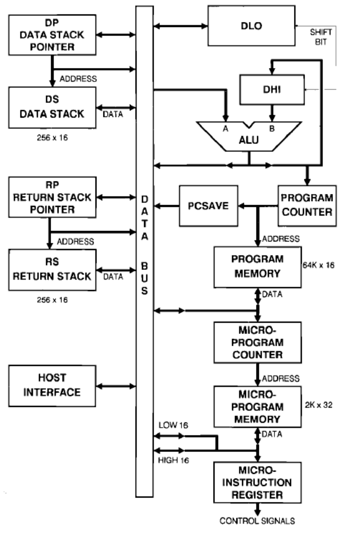
Claim Language	
	As a further example, the KDF9 architecture, which was developed in the 1960s, clearly illustrates a system that includes a specific register and incrementer as a loop counter: Any main store reference has associated with it a Q-store number (or an implied Q0), and refers to the address specified, augmented by the content of the modifier section. If the letter Q is added to the mnemonic form of the instruction then after the address has been calculated the modifier is changed by addition of the increment and the counter is reduced by one. Jump instructions testing the counters are, of course, provided. [HTCTP10218055, the KDF9 Reference] A person of ordinary skill in the art would have been motivated to combine the KDF9 Q-store concept with the Transputer Architecture for this limitation because, for example, it would potentially increase the efficiency of loop execution.
said loop counter being connected to a decrementer,	The Transputer Architecture discloses or renders obvious this limitation as any memory location can serve as a loop counter if a specific register as a loop counter is not required. Furthermore, the Transputer discloses specific instructions for loop control, which indicates the existence of a loop counter connected to a decrementer. As shown in Figure 4.15, the Transputer Reference Manual discloses a specific instruction to control looping:  Table 4.15: IMS T414 control operation codes [HTCTP0111984] This is further explained in the Transputer Instruction Set: A Compiler Writer's Guide

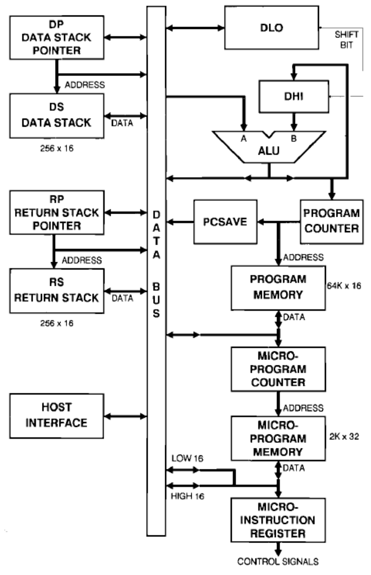
Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DE
20	22F0	ret	5	return	
1B	21FB	ldpi	2	load pointer to instruction	
3C	23FC	gajw	2	general adjust workspace	
06	F6	gcall	4	general call	
21	22F1	lend	10	loop end (loop)	D
			5	loop end (exit)	D

Claim Language	
	which explains: *lend* will decrement the iteration count and, if the number of iterations remaining is greater than zero, increment the control variable and subtract **Areg** from **Iptr**. If the number of iterations left after the decrement is less than or equal to zero then execution passes to the next instruction. [HTCTP10218113] Additionally, the A register of the Transputer can be used as a loop counter. It can be decremented by an instruction such as ADC -1 and tested using the CJ instruction. [*See, e.g.*, HTCTP10218105, HTCTP10218098]. Even if this claim element requires a specific register as a loop counter, such architecture was well known by a person in the art at the time of the invention. For example, the Koopman reference discloses a specific register that may be used as a loop counter. [*See*, e.g., HTCTP10218898-899].

Claim Language	
	 CONTEXT REGISTER OUTPUT LATCH I/O BUS (8 BITS) LASTX X Y Z A B ALU SHIFTER CONTROL LOGIC INSTRUCTION CACHE (6 REGISTERS) DATA 16 BITS PROGRAM MEMORY ADDRESS INSTRUCTION POINTER INDEX DATA STACK POINTER RETURN STACK POINTER Fig. 4.4 — M17 block diagram. [HTCTP10218898] The top element of the Return Stack is buffered in the INDEX register. The INDEX register doubles as a count-down counter for use in program loops and the instruction repeat feature. [HTCTP10218899] A person of ordinary skill in the art would have been motivated to combine the Koopman reference with the architecture with the Transputer Architecture for this limitation because, for example, it would potentially increase the efficiency of loop execution.

Claim Language	
	As a further example, the KDF9 architecture, which was developed in the 1960s, clearly illustrates a system that includes a specific register and incrementer as a loop counter: Any main store reference has associated with it a Q-store number (or an implied Q0), and refers to the address specified, augmented by the content of the modifier section. If the letter Q is added to the mnemonic form of the instruction then after the address has been calculated the modifier is changed by addition of the increment and the counter is reduced by one. Jump instructions testing the counters are, of course, provided. [HTCTP10218055] A person of ordinary skill in the art would have been motivated to combine the KDF9 Q-store concept with the Transputer Architecture for this limitation because, for example, it would potentially increase the efficiency of loop execution.
said internal data bus being bidirectionally connected to a stack pointer, return stack pointer, mode register and instruction register,	The Transputer Architecture discloses or renders obvious this limitation. For example, the Transputer includes an internal set of data buses.

Claim Language	
	[’890 Patent, Fig. 2] For example, as shown below, Figure 4.1 of Koopman discloses a return stack pointer. A person of ordinary skill in the art would have been motivated to combine the Koopman reference with the Transputer Architecture for this limitation if, for example, the person is trying to implement a single chip computer with integrated communications suitable for executing stack based processing languages such as FORTH.  DP DATA STACK POINTER ADDRESS DS DATA STACK DATA 256 x 16 RP RETURN STACK POINTER ADDRESS RS RETURN STACK DATA 256 x 16 HOST INTERFACE DATA BUS DLO SHIFT BIT DHI A B ALU PCSAVE PROGRAM COUNTER ADDRESS PROGRAM MEMORY 64K x 16 DATA MICRO-PROGRAM COUNTER ADDRESS MICRO-PROGRAM MEMORY 2K x 32 DATA LOW 16 HIGH 16 MICRO-INSTRUCTION REGISTER CONTROL SIGNALS Fig. 4.1 — CPU/16 block diagram. [HTCTP10218888]
said stack pointer pointing into said first push down stack,	The Transputer Architecture discloses or renders obvious this limitation. For example, the Transputer Reference Manual’s disclosure of the ‘workspace pointer’ (as discussed above) is a stack pointer.

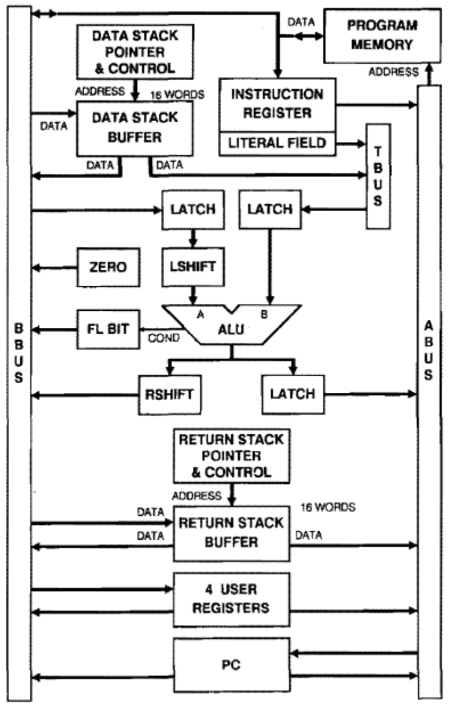
Claim Language	
	includes a three register floating-point evaluation stack, containing the AF, BF, and CF registers. When values are loaded onto, or stored from the stack the AF, BF and CF registers push and pop in the same way as the A, B and C registers. [HTCTP0111889] Additionally, it would have been obvious to one of skill in the art to use a stack pointer pointing to a push down stack. Stack pointers had been used for many years (for example in the PDP11, M68000 and i8086) to provide stacks in memory both for procedure calling and for expression evaluation. Furthermore, to the extent this claim element requires the stack pointer to directly address the register stack; it would have been obvious to one skilled in the art to implement such a design. For example, the Koopman reference discloses such architecture as shown below:  DP DATA STACK POINTER ADDRESS DS DATA STACK DATA 256 x 16 RP RETURN STACK POINTER ADDRESS RS RETURN STACK DATA 256 x 16 HOST INTERFACE DATA BUS DLO SHIFT BIT DHI A B ALU PCSAVE PROGRAM COUNTER ADDRESS PROGRAM MEMORY 64K x 16 DATA MICRO-PROGRAM COUNTER ADDRESS MICRO-PROGRAM MEMORY 2K x 32 DATA LOW 16 HIGH 16 MICRO-INSTRUCTION REGISTER CONTROL SIGNALS Fig. 4.1 — CPU/16 block diagram.

Claim Language	
	[HTCTP10218888] A person of ordinary skill in the art would have been motivated to combine the stack pointer and registers of Koopman with the Transputer Architecture for this limitation if, for example, the person is trying to implement a single chip computer with integrated communications suitable for executing stack based processing languages such as FORTH. Additionally, U.S. Patent Nos. 4,153,933, Blume Jr. et al., issued May 8, 1979, discloses a stack pointer pointing to a push down stack. [*See*, *e.g.*, HTCTP10218681, HTCTP10218685, U.S. Patent No. 4,153,933: col. 3, lines 53-58]. One of ordinary skill in the art would have been motivated to combine the Transputer Reference Manual with the ’933 Patent to obtain a microprocessor having a stack pointer pointing into a push down stack, as doing so might improve microprocessor performance and/or reduce cost.
said internal data bus being connected to a memory controller, to a Y register of a return push down stack, an X register and a program counter,	The Transputer Architecture discloses or renders obvious this limitation. For example the Transputer Reference discloses the internal data and address buses connected to an external memory controller.

Claim Language	
	Since a program requires less store to represent it, less of the memory bandwidth is taken up with fetching instructions. Furthermore, as memory is word accessed the processor will receive four instructions for every fetch. [HTCTP0111912]
Claim 17	
17. The microprocessor of claim 11 additionally comprising	See Claim 11 above
A ring oscillator variable speed system clock connected to said main central processing unit and said ring oscillator variable speed system clock being provided in a single integrated circuit.	The Court has construed the claim term “ring oscillator” as “an oscillator having multiple, odd number of inversions arranged in a loop, wherein the oscillator is variable based on the temperature, voltage and process parameters in the environment.” Under the Court’s construction, and HTC’s construction, it would have been obvious to one skilled in the art at the time of the invention to attempt to combine the Transputer with a “ring oscillator variable speed system clock,” to avoid the need of constructing an on-chip Phase Locked Loop. This type of system was disclosed in Introduction to VLSI Systems by Mead & Conway. [*See* HTCTP10219186 - HTCTP10219618]. The motivation would have been to enable high speed internal operation together with low-speed external interfaces. Process variation in integrated circuit fabrication does not allow accurate resonant networks to be fabricated by usual means, but it is perfectly feasible, indeed essential for self-contained VLSI systems, to generate clock signals on the chip. [HTCTP10219451] The easiest way to build these timers is as chains of inverters. The propagation delay time of such a chain will of course vary with tau, according to the way in which the fabrication process, aging, temperature,

Claim Language	
	and power voltage affect tau. [HTCTP10219452] However, these variations only make the inverter chain a better model of the system being clocked than a fixed timer would be, since on the same piece of silicon these variable factors are nearly the same for the clock and for the system. [HTCTP10219452] Clocks that employ these delays as timers are all elaborations of the *ring oscillator* circuit shown in Fig. 7.9(a). Rings of an odd number of inverters have no stable condition and will oscillate with a period that is some odd submultiple of the delay time twice around the ring. [HTCTP10219453] Additionally, U.S. Patent No. 4,691,124 (“Ledzius”) discloses a “ring oscillator variable speed system clock” based on the Court’s construction of “ring oscillator.” Ledzius describes a variable frequency oscillator whose oscillation frequency is intended to match the timing of on-chip logic circuits under varying environmental conditions. An odd number of inversions arranged in a loop can be identified in this variable frequency oscillator whose speed depends on environmental parameters. It can be stopped and started to allow synchronization with an external interface. Among other things, the speed of the clock is determined by the delay element 52, shown below, which is intended to track the speed of the other components of the system. This clock is used to time the operations of the logic circuits which could be a microprocessor. In this case, I believe that it would be obvious to try to apply the Ledzius technique to the Transputer design. The motivation would have been to enable high speed internal operation together with low-speed external interfaces.

Claim Language	
	[HTCTP0111985] The Transputer CPU clock generator is described in the Transputer Reference Manual. [*See, e.g.*, HTCTP0111985] The Transputer had a standard 5MHz frequency as an input. It was used as a reference by the internal PLL clock generators to provide high speed internal clocks. This is further described by the Transputer Patents. For example, U.S. Patent No. 4,689,581 (“Talbot”) filed on May 9, 1985 and published on August 25, 1987 [HTCTP0003194-205]. Talbot discloses a PLL clock for a Transputer CPU that was programmed during manufacture to run at a multiple of the 5MHz reference frequency after testing to determine the highest speed the CPU could operate. [*See, e.g.*, HTCTP0111993, HTCTP0112206; see also HTCTP0003200, Talbot Col 3, Lns. 52 - 58]. This patent was one of the first disclosures of a PLL clock generator for a microprocessor.
Claim 19	
19. The microprocessor of claim 11	See Claim 11 above
in which said first push down stack has a first plurality of stack elements configured as latches,	The Transputer Architecture discloses or renders obvious this limitation. For example, the A, B, and C registers disclosed in the Transputer Reference Manual are configured as latches. The A, B and C registers which form an evaluation stack, and are the sources and destinations for most arithmetic and logical operations. Loading a value into the stack pushes B into C, and A into B, before loading A. Storing a value from A, pops B into A and C into B. [HTCTP0111890]
a second plurality of stack elements configured as a random access memory, said first and second	The Transputer Architecture discloses or renders obvious this limitation. For example, the Transputer has A, B, and C registers that can be considered the “first plurality.” The internal Transputer memory can be used to provide a “second plurality,” and an external

Claim Language	
plurality of stack elements and said central processing unit being provided in a single integrated circuit, and a third plurality of stack elements configured as a random access memory external to said single integrated circuit.	memory can be used to provide a “third plurality.” [*See*, *e.g.*, HTCTP0111913]. Furthermore, such architecture was well known in the art at the time of the ’890 patent. For example, the Koopman reference discloses such a system:  DATA STACK POINTER & CONTROL DATA PROGRAM MEMORY ADDRESS ADDRESS 16 WORDS INSTRUCTION REGISTER LITERAL FIELD DATA DATA STACK BUFFER DATA DATA T BUS LATCH LATCH ZERO LSHIFT A B ALU COND FL BIT B BUS A BUS RSHIFT LATCH RETURN STACK POINTER & CONTROL ADDRESS DATA 16 WORDS RETURN STACK BUFFER DATA DATA 4 USER REGISTERS PC Fig. 5.1 — FRISC 3 block diagram. [HTCTP10218925] One of the innovative features of the FRISC 3 is the use of stack management logic associated with the stack pointers. This logic

Claim Language	
	automatically moves stack items between the 16-word on-chip stacks and a program memory stack spilling area to guarantee that the on-chip stack buffers never experience an overflow or underflow. This logic steals program memory cycles from the processor to accomplish this, avoiding the extra stack data pins on the chip in exchange for a small performance degradation spread throughout program execution. The designers of the FRISC 3 call this feature a stack cache, because it caches the top few stack elements for quick access on-chip. This cache is not like normal data or instruction caches in that it does not employ an associative memory lookup structure to allow access to data residing in scattered areas of memory. [HTCTP10218926] A person of ordinary skill in the art would have been motivated to combine the “stack management logic” of the Koopman reference with the Transputer Architecture in trying to implement a single chip computer with integrated communications suitable for executing stack based processing languages such as FORTH.